Global HPCC Benchmarks in Chapel

Brad Chamberlain, Steve Deitz, Samuel Figueroa, David Iten, Andy Stone*

Cray Inc., *Colorado State University

SC08: HPC Challenge BOF
November 18, 2008
Chapel in a nutshell

**Chapel:**
- a new parallel language being developed by Cray Inc.
- part of DARPA’s HPCS* program
- first public release occurred this past weekend

*HPCS = High Productivity Computing Systems*
When we last saw you at HPCC...

**HPCC 2006**: Chapel “elegance only” entry

- **goal**: show where Chapel was headed
- 3 benchmarks: STREAM Triad, Random Access, FFT
- written with elegance and scalability in mind
- compiled and executed correctly, *but*:
  - only supported single-threaded execution
  - leaked memory
  - =>no performance
This year’s entry

- First public performance numbers for Chapel execution
- First distributed memory execution of our data parallel features
- As intended, our code is quite similar to 2006 entry
- First locality-sensitive implementation of HPL in Chapel

Please set your expectations appropriately:

- This is a snapshot of a work in progress, not the final word
- Our first distribution ran for the first time only two months ago

Focus less on our current performance and more on how we got it
Chapel is 3.7 – 50× shorter; yet this is not a fair comparison due to multiple algorithms in reference versions.
STREAM Triad in Chapel

```chapel
const BlockDist = new Block1D(bbox=[1..m], tasksPerLocale=...);

const ProblemSpace: domain(1, int(64)) distributed BlockDist = [1..m];

var A, B, C: [ProblemSpace] real;

forall (a, b, c) in (A, B, C) do
  a = b + alpha * c;
```

Chapel (7)
Chapel Distributions

*Distributions:* “Recipes for parallel, distributed arrays”
- help the compiler map from the computation’s global view...

...down to the *fragmented*, per-processor implementation
Chapel Distributions

- (Advanced) Programmers can write distributions in Chapel
- Chapel will support a standard library of distributions
  - *research goal*: using the same mechanism that users would
- Block1D is our first such distribution
  - *our compiler has no semantic knowledge of block distributions*
  - only of a distribution’s interface—how to…
    - …create domains and arrays using that distribution
    - …map indices to locales
    - …access array elements
    - …iterate over indices/array elements
      - sequentially
      - in parallel
      - in parallel and zippered with other parallel iterable types
    - …and so forth…

Chapel (9)
# Experimental Platform

<table>
<thead>
<tr>
<th>machine characteristic</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>jaguar</td>
</tr>
<tr>
<td>model</td>
<td>Cray XT4</td>
</tr>
<tr>
<td>location</td>
<td>ORNL</td>
</tr>
<tr>
<td># compute nodes</td>
<td>7,832</td>
</tr>
<tr>
<td>compute node processor</td>
<td>2.1 GHz AMD Opteron</td>
</tr>
<tr>
<td>cores per node</td>
<td>4</td>
</tr>
<tr>
<td>total user RAM per node</td>
<td>7.68 GB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STREAM Triad characteristic</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>per-node problem size</td>
<td>85,985,408</td>
</tr>
<tr>
<td>per-node memory required</td>
<td>1.92 GB</td>
</tr>
<tr>
<td>percent of available memory</td>
<td>25.0%</td>
</tr>
</tbody>
</table>
Chapel STREAM Performance

STREAM Triad Performance (in GFlop/s)

Reference: 1 proc/locale extrapolated
Chapel: 1 task/locale
Chapel: 2 tasks/locale
Chapel: 3 tasks/locale
Chapel: 4 tasks/locale
Chapel: 5 tasks/locale

1.69 TFlop/s
Why doesn’t Chapel scale perfectly?

- Because Block1D’s current parallel iterator is very naive…

- Ditto for its termination…
Strategies for improvement

- Use tree-based startup/teardown to convert $O(p)$ to $O(\lg p)$

- Or: Have compiler optimize code to use SPMD exec. model
  - reduces $O(\lg p)$ to $O(1)$ by amortizing into program startup/teardown
SPMD-style Chapel

- In the meantime, users can code in SPMD like the MPI version using Chapel’s support for *multiresolution programming*:

```chapel
var localGBs: [LocaleSpace] real;

tlocal GBs: [LocaleSpace] real;

coforall loc inLocales do
on loc {
const myProblemSpace: domain(1, int(64))
    = BlockPartition(ProblemSpace, here.id, numLocales);
var myA, myB, myC: [myProblemSpace] real(64);
const startTime = getCurrentTime();
local {
for (a, b, c) in (myA, myB, myC) do
    a = b + alpha * c;
}
const execTime = getCurrentTime() - startTime;
localGBs(here.id) = timeToGFlops(execTime);
}

const avgGBs = (+ reduce localGBs) / numLocales;
```

Chapel (14)
SPMD Chapel Performance

STREAM Triad Performance (GFlop/s per locale)

- Reference: 4 MPI procs/locale
- Reference: 1 MPI proc x 4 OpenMP threads/locale
- Chapel: 4 tasks/locale
- Reference: 1 MPI proc/locale
- Chapel: 1 task/locale

Number of Locales
RA Declarations in Chapel

```chapel
class
class RA

const TableDist  = new Block1D(bbox=[0..m-1], tasksPerLocale=...),
UpdateDist = new Block1D(bbox=[0..N_U-1], tasksPerLocale=...);

const TableSpace: domain(1, uint(64)) distributed TableDist = [0..m-1],
Updates: domain(1, uint(64)) distributed UpdateDist = [0..N_U-1];

var T: [TableSpace] uint(64);
```
RA Computation in Chapel

```chapel
const TableSpace: domain(1, uint(64)) distributed TableDist = [0..m-1],
  Updates: domain(1, uint(64)) distributed UpdateDist = [0..N_U-1];

var T: [TableSpace] uint(64);

forall (_, r) in (Updates, RAStream()) do
on T(r&indexMask) do
  T(r&indexMask) ^= r;
```

![Diagram showing the RA Computation process](Diagram.png)
RA Performance in Chapel

RA Performance (in GUPS)

Chapel (1 task/locale)

Number of Locales

0.0012
0.001
0.0008
0.0006
0.0004
0.0002
0

1 4 8 16 32 64
FFT and HPL Status

- **FFT:**
  - not yet running on distributed memory
    - Block1D not yet rich enough to support slicing, re-indexing
  - have made a big effort to reclaim descriptor memory from slicing
    - can now run full problem size

- **HPL:**
  - not yet running on distributed memory
    - need to add block-cyclic, dimensional, and replicated distributions
  - current version written to be locality-aware

- All four of these codes are very clean and should serve as great references to others attempting the HPC Challenge
Acknowledgements

This material is based upon work supported by the Defense Advanced Research Projects Agency under its Agreement No. HR0011-07-9-0001.

This research used resources of the National Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725

Thanks also to our many colleagues who have helped us reach this point!
Summary

- Chapel is scaling on dist. memory machines, if not perfectly
  - more importantly, scalability limiters are known and addressable
- Chapel achieved its first TeraFlop
- Chapel has started to demonstrate user-defined distributions
  - Recall that these have only been working for two months
  - (and a busy two months at that: first public release, two tutorials, …)
- See you at HPCC 2009!

In the meantime, download Chapel, try it out, and please give us your feedback:

http://chapel.cs.washington.edu

(our HPCC codes and report are available within the release)